

Methods and Implementations for Integer Factorization



Dana Jacobsen
CS567 Cryptology I
16 December 2009

Integer Factorization

- Integers (not polynomials)
- Any n , p 's prime: $n = p_1 \cdot p_2 \cdot p_3 \cdot \dots \cdot p_k$
- Easy to verify solution, hard to do
- No polynomial algorithm known (excluding quantum computers)

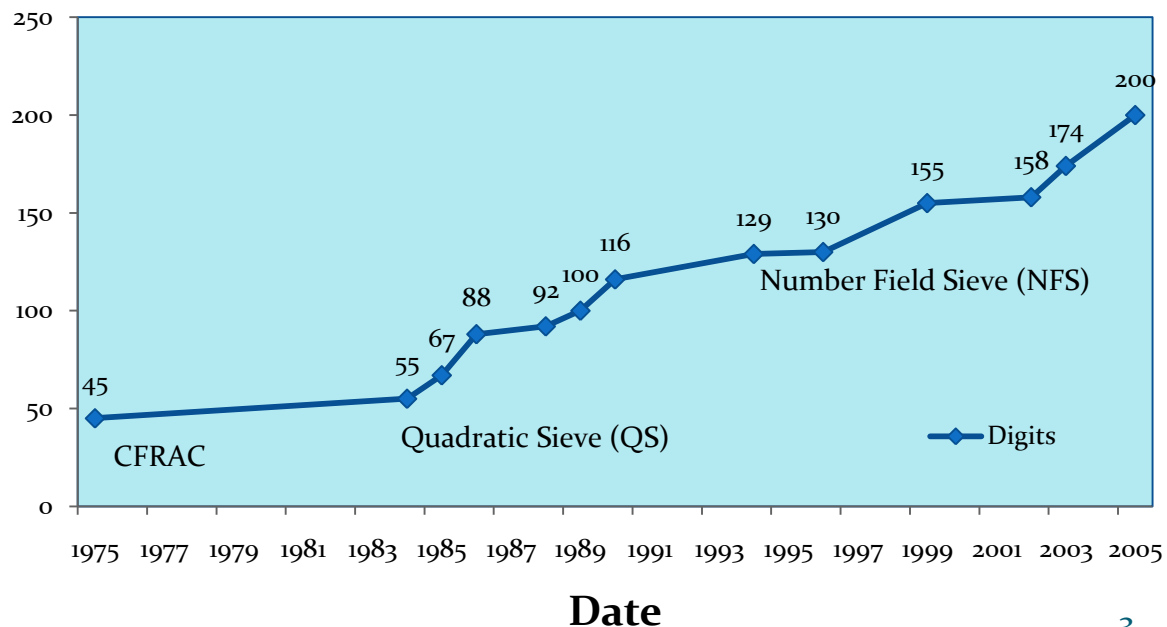
Why do we care?

- RSA Encryption based on factoring the product of two large primes being hard

- What is large?

- 1978: 512b (174d)
- 1980s: 768b (231d)
- 1990s: 1024b (309d)

Factorization Records (Digits)



Just large numbers?

- Surprisingly, no!
- Quadratic Sieve and Number Field Sieve involve factoring millions of small (~ 64 -bit) numbers
- Efficiency for small numbers is important

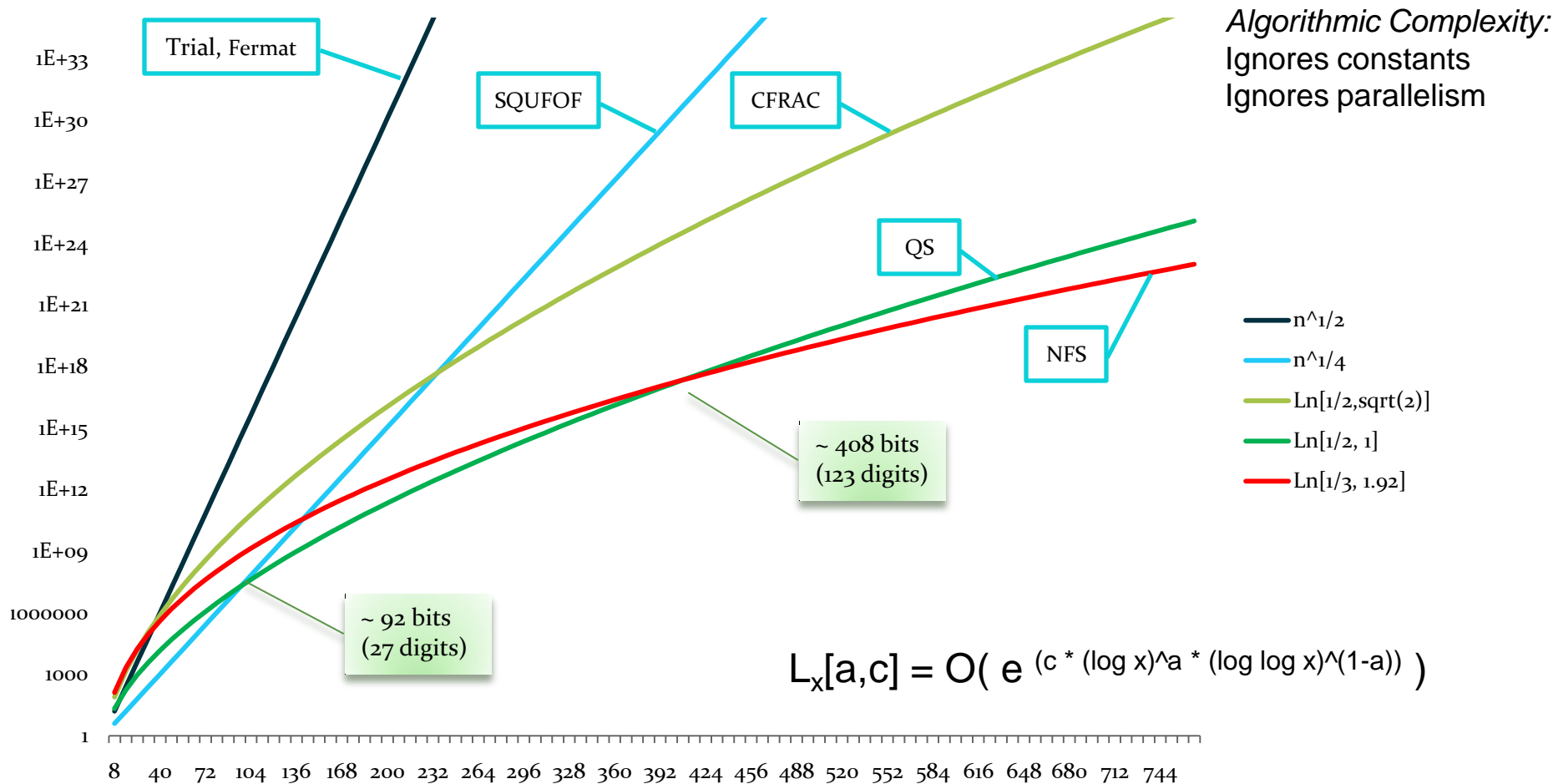
Methods vs. Implementations

- **Methods: Algorithms**
 - Fermat's Method
 - Elliptical Curve Method
 - Quadratic Sieve
 - etc.
- **Implementations: Code**
 - DAJ (my implementations)
 - Msieve
 - YAFU
 - FLINT
 - Pari
 - etc.

Methods

Name	Inventor	Date	Complexity	Depends on
Trial Division			$P \sim N^{1/2}$	Size of p
Fermat	D. Fermat	Circa 1650	$N^{1/2}$	Size of N
SQUFOF	D. Shanks	1971	$N^{1/4}$	Size of N
Lehman (Fermat)	R. Lehman	1974	$N^{1/3}$	Size of N
$P-1$ $P+1$	J. Pollard H. Williams	1974 1982	$\text{Blog}B$	Smoothness of factor
Pollard's Rho	J. Pollard	1975	$P^{1/2}$	Monte carlo
Continued Fractions	Brillhart, Morrison	1975*	$\text{Ln}[1/2, \text{sqrt}(2)]$	Size of N
ECM	H. Lenstra	1987	$\text{Lp}[1/2, \text{sqrt}(2)]$	Size of p
Dixon's	J. Dixon	1981	$\text{Ln}[1/2, 2 \text{sqrt}(2)]$	Size of N
Quadratic Sieve	C. Pomerance	1981 (1985)	$\text{Ln}[1/2, 1]$	Size of N
MPQS	R. Silverman	1987	$\text{Ln}[1/2, 1]$	Size of N
SIQS	P. Montgomery	1993	$\text{Ln}[1/2, 1]$	Size of N
Number Field Sieve (Special NFS)	J. Pollard	1993 1988	$\text{Ln}[1/3, 1.92]$ $\text{Ln}[1/3, 1.52]$	Size of N

Complexity Overview



Implementations

- 32-bit (10 digit), 64-bit (19 digit), GMP, other
- Difficulty of Implementation
 - Simple
 - Trial, Fermat, SQUFOF, Pollard's Rho, Pollard's $p-1$, mix.
 - ECM and QS: Effort to make it really fast
 - NFS: Very difficult

DAJ (my code)

Version	
Source	On request from dana@acm.org
What is it?	64-bit and GMP routines for simple methods
Methods	Trial, Rho, Rho (Brent), P-1, SQUFOF (Msieve and yafu variants in 64-bit and GMP), Fermat, HOLF, P-1/P+1/ECM via GMP-ECM
Mix is	Trial, SQUFOF, Rho, P-1, Fermat, HOLF, P-1/P+1/ECM
Thoughts	Some methods not generally seen 64-bit and GMP versions Without QS, runs out of steam quickly

MIRACL

Version	5.4
Source	http://www.shamus.ie
What is it?	Complete BigNum package
Methods	Trial, SQUFOF (racing 62-bit), Rho, P-1/P+1/ECM via GMP-ECM, tinyQS, MPQS, NFS
Mix is	Trial, Rho/Brent, P+1, P-1, ECM, MPQS
Thoughts	Some nice examples Generally slow

PARI/GP

Version	2.3.4 (latest is 2.4.3alpha)
Source	http://pari.math.u-bordeaux.fr/
What is it?	Computer algebra system
Methods	Rho, SQUFOF, ECM, MPQS.
Mix is	Trial, Powers, Rho (Brent), SQUFOF, ECM, MPQS, more ECM
Thoughts	Some of the most sophisticated code for small numbers MPQS not well developed, good to about 47 digits

FLINT

Version	1.6
Source	http://www.flintlib.org/
What is it?	Computer algebra library
Methods	Trial, HOLF, SQUFOF, QS, MPQS
Mix is	Trial, Powers, HOLF, tinyQS, SQUFOF, more trial Alternate: QS, MPQS
Thoughts	Meant as a programming library for computer algebra. Concentrates more on polynomial logic Example factoring code brittle and most 64-bit only MPQS competitive until about 55 digits

Msieve

Version	1.43
Source	http://sourceforge.net/projects/msieve/
What is it?	Complete factoring package, all public domain
Methods	Trial, SQUFOF (racing 62-bit), Rho, P-1/P+1/ECM via GMP-ECM, tinyQS (≤ 85 bits), MPQS (85+ bits), NFS (277+ bits)
Mix is	Trial, Rho, ≤ 60 bits: SQUFOF, QS ≤ 85 bits: QS else: adaptive P-1, P+1, ECM, Powers, MPQS
Thoughts	All around good choice Back-end used by many other packages Best in combination with GGNFS Not well threaded NFS line siever is painfully slow GPU version of NFS poly selection

YAFU

Version	1.12
Source	http://sites.google.com/site/bbuhrow/home
What is it?	Yet Another Factorization Utility
Methods	Trial, SQUFOF (62-bit), Rho, P-1, P+1, ECM, QS (40-177 bits), MPQS (60-255 bits), SIQS (150 bits, max 150 digits)
Mix is	Trial, Rho (Brent), <= 99 bits: QS <= 135 bits: MPQS <= 160 bits: P+1, SIQS <= 180 bits: P+1, P-1, SIQS else: P+1, P-1, adaptive ECM, SIQS
Thoughts	Borrows extensively from Msieve SIQS is well threaded and very fast Fastest for medium size (50-100 digits) A bit less polished than Msieve

GGNFS

Version	0.77.1
Source	http://sourceforge.net/projects/ggnfs/
What is it?	General Number Field Sieve
Methods	GNFS, SNFS
Mix is	N/A
Thoughts	Turnkey NFS – very easy to use Franke/Kleinjung Lattice Siever -- FAST Integrates with Msieve Very easy to run in parallel

Others

- MAPLE (Prho, ECM, CFRAC, SQUFOF, MPQS)
- Kechlibar's MPQS/SIQS (hand parameters)
- Alpern's JAVA ECM and SIQS (easy to run, decent speed)
- Scott Contini MPQS/SIQS examples
- CADO NFS (lots of hand parameters)
- pGNFS
- kmGNFS

Hardware Details

- CPU: Intel Q6600 running at 3.6GHz
- GPU: NVIDIA GTX260 core 216
- Memory: 8GB DDR2
- Disk: 3TB SATA RAID-5 (Linux MD)
- O/S: Linux x64 (Fedora 11), gcc 4.1.4

How fast can we factor?

Digits of semiprime factored in the time limit

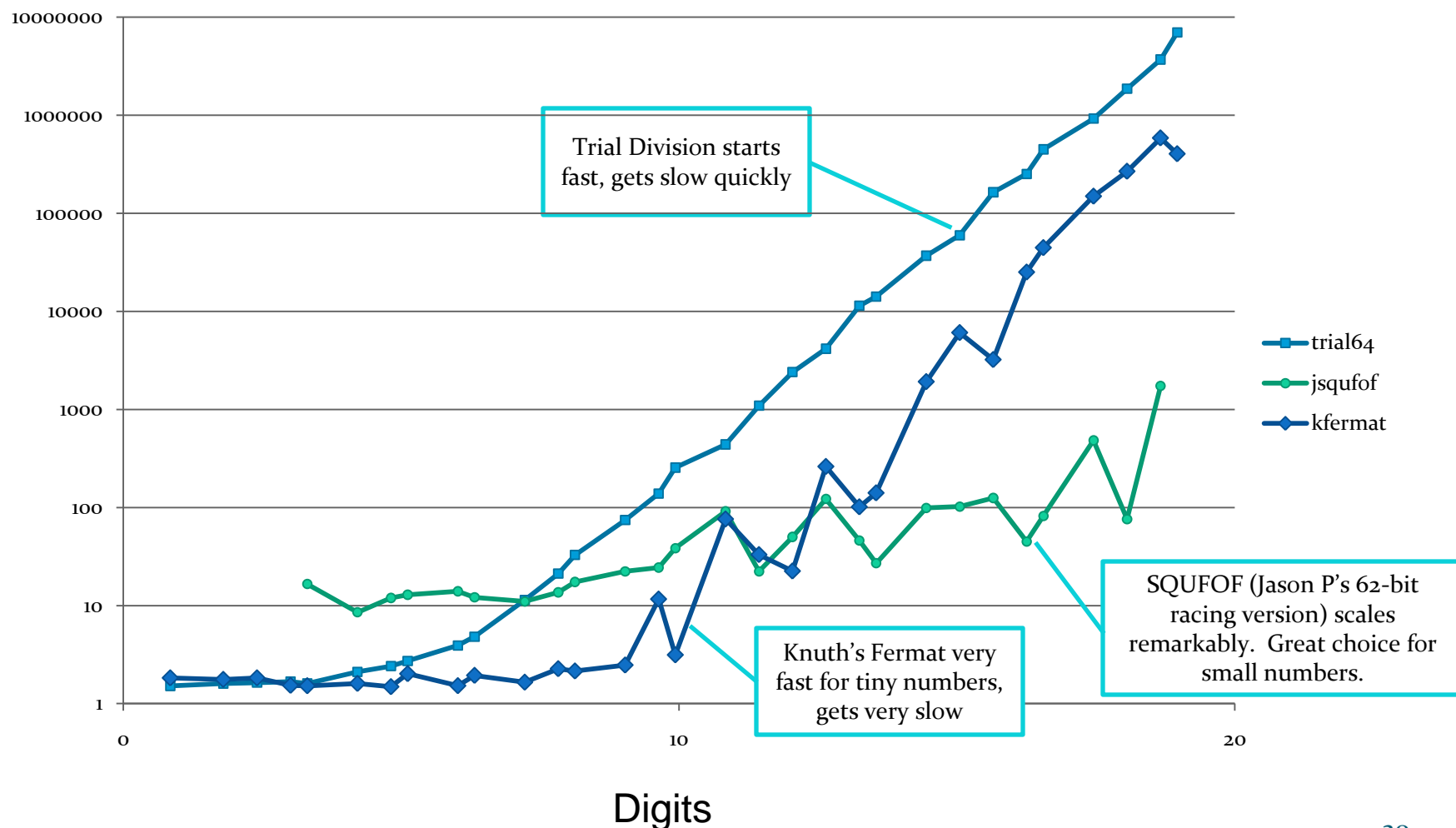
Name	0.02 s	1 second	1 minute	1 hour
Trial Division	15	18	21	24
Fermat	17	19	23	27
Pollard's Rho	18	25	33	39
SQUFOF	19	25	32	40
MPQS	32 (YAFU) 33 (FLINT) 26 (Msieve)	50 (YAFU) 52 (FLINT) 54 (Msieve)	68 (YAFU) 70 (FLINT) 71 (Msieve)	80 (FLINT) 92 (Msieve)
SIQS	22 (YAFU)	55 (YAFU)	77 (YAFU)	97 (YAFU)
Number Field Sieve				98 (GGNFS+ms)
MIX	25 (MIRACL) 26 (Msieve) 32 (PARI) -- (YAFU)	35 (MIRACL) 48 (PARI) 53 (Msieve) 50 (YAFU)	62 (MIRACL) 66 (PARI) 71 (Msieve) 76 (YAFU)	75 (MIRACL) 83 (PARI) 91 (Msieve) 97 (YAFU)

Special Cases

- Small Factors:
 - Trial Division
 - Pollard Rho
 - ECM
- Smooth $P-1$ or $P+1$
- Exact Powers ($n=p^k$)
- p and q very close: Fermat
- $p/q = u/v$: Lehman's pre-multiplier for Fermat
- Hart's One Line Factorization
- Special Number Field Sieve (1.523 vs. 1.92)

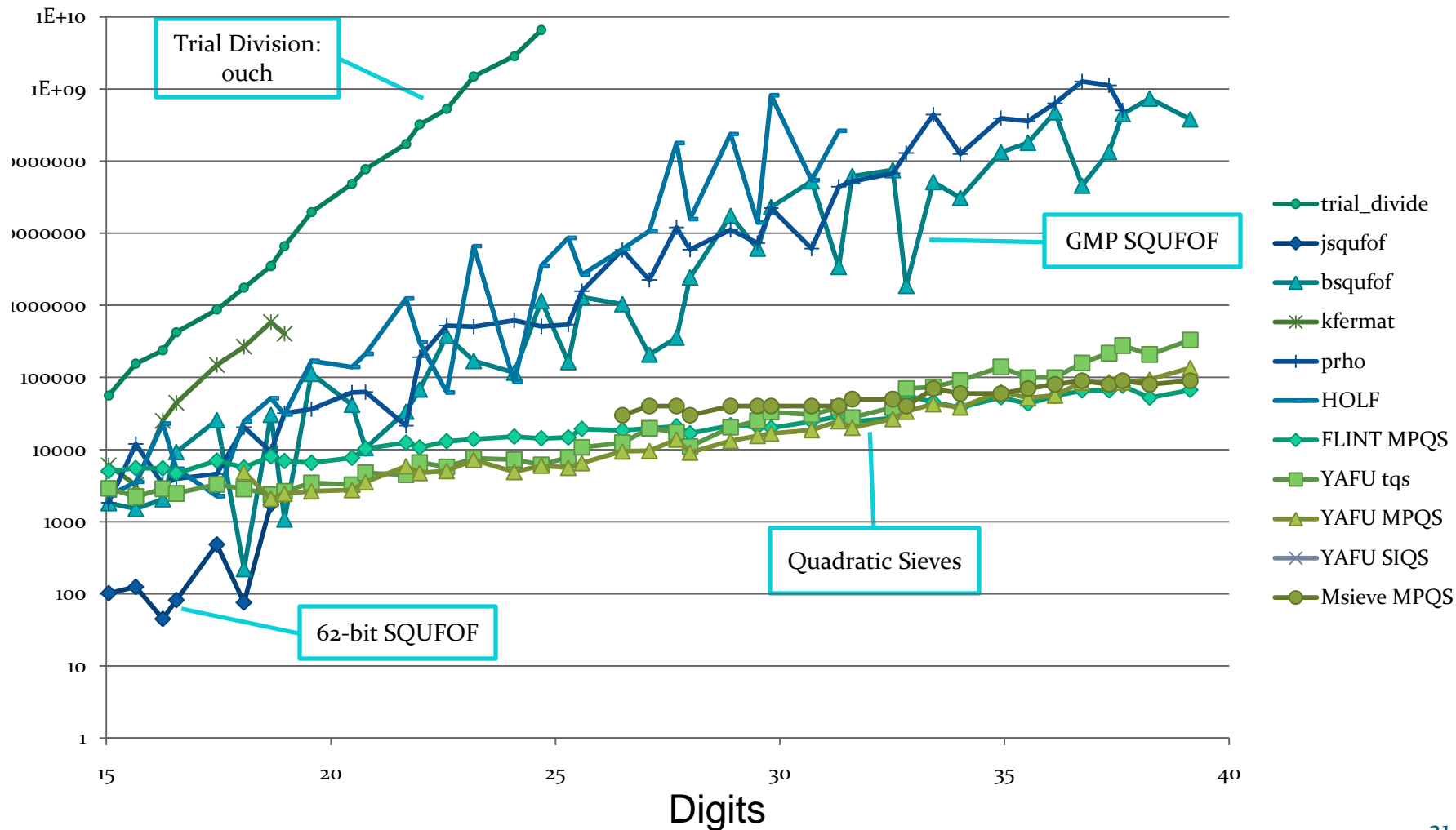
Measured Speeds (<20 digits)

Time in microseconds to factor a semiprime



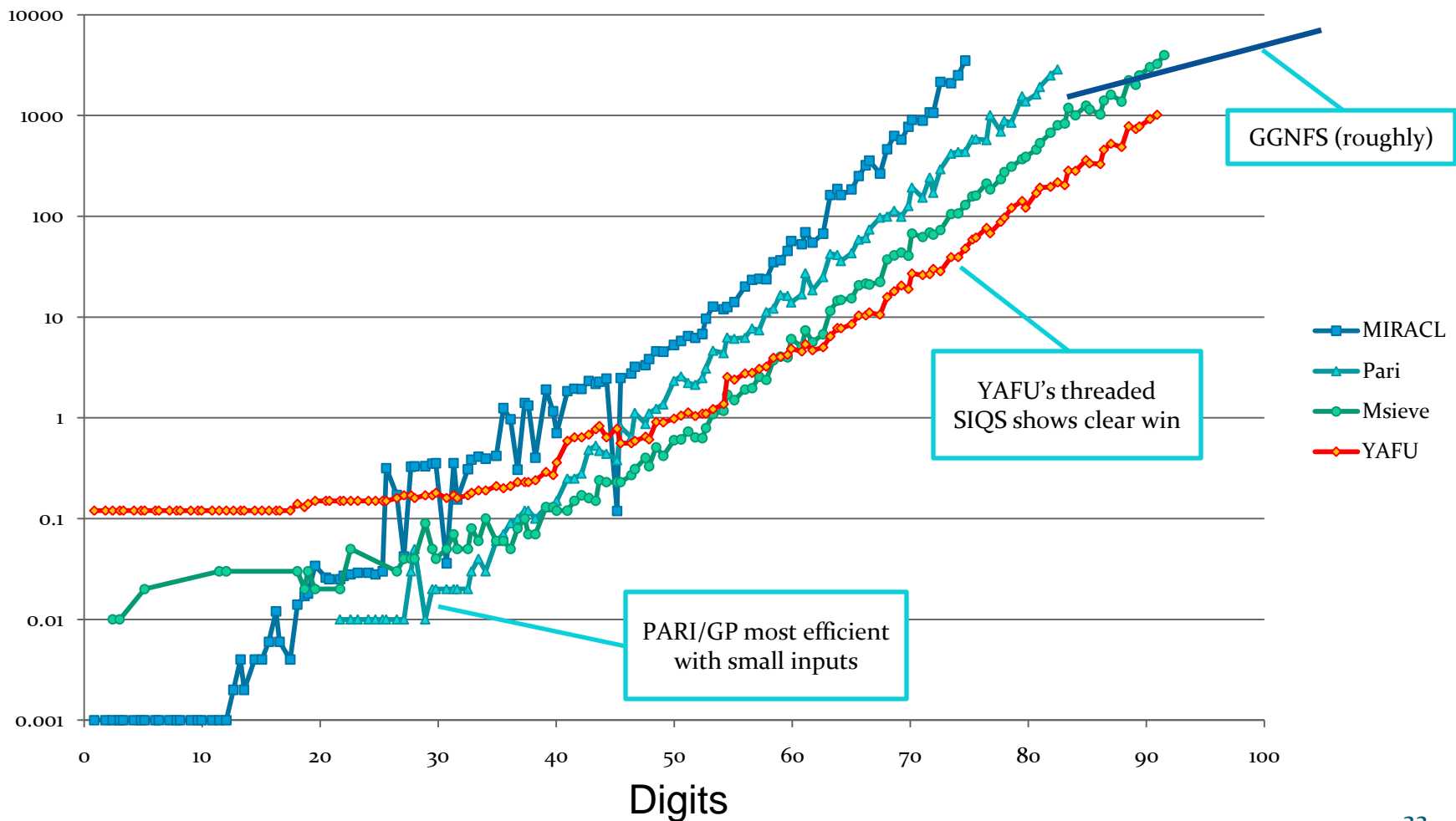
Measured Speeds (15-40 digits)

Time in microseconds to factor a semiprime



Measured Speeds (general)

Time in seconds to factor a semiprime



Fastest Methods

- ≤ 10 digits: trial division, Fermat's (Knuth)
 - 11-18 digits: SQUFOF (racing 62-bit)
 - 19-100 digits: Quadratic Sieve
 - 100+ digits: NFS
-
- Pari, Msieve, YAFU, GGNFS+Msieve

Combining GGNFS and Msieve

- Polynomial selection:
 - GGNFS pol₅₁ (Kleinjung and Franke)
 - Msieve CPU
 - Msieve GPU (beta, 10-30x faster than CPU)
- Sieving:
 - GGNFS (Franke/Kleinjung lattice siever)
 - Msieve (line siever)
- Linear Algebra:
 - GGNFS
 - Msieve

Combining GGNFS and Msieve

- Polynomial Selection: Either
- Sieving: GGNFS by 100x
- Linear Algebra: Msieve

- Use the `factMsieve.pl` script!

- 122 digit semiprime: 6.5h poly, 3h sieve on 5 quad-cores, 0.5h linear algebra: ~10 hours total

Discussion

- What is the optimal hardware / software setup?
 - Lots of fast cores
 - YAFU for simple factoring
 - GGNFS + Msieve, use factMsieve.pl script
 - GPU for polynomial selection? Opportunity for software enhancements here
- Are some numbers easier to factor?
 - Small factors (trial, rho, ECM)
 - Smooth ($p-1$, $p+1$)
 - p/q close (Fermat)
 - $p/q = u/v$ (Hart's)
 - Special
- What is the most time consuming step?
 - Typically sieving unless one has many cores, then it is polynomial selection and back-end.

Questions